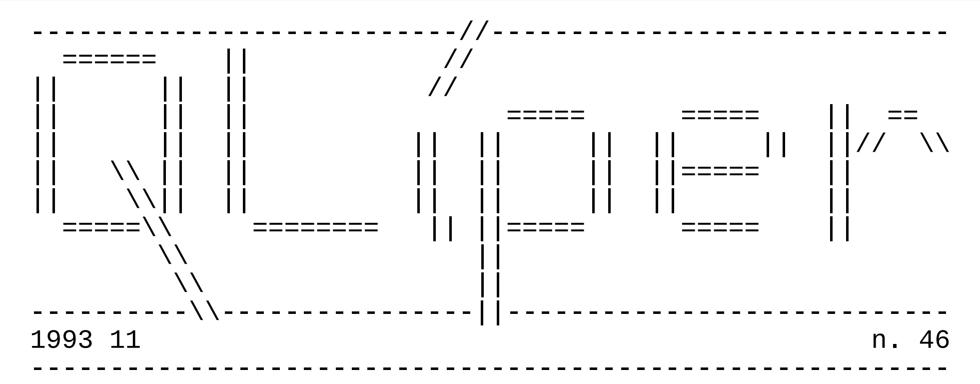


RE-PRESENTACIóN
===============

Creo que estamos muy parados, yo el primero. No voy a volver a lloriquear por el retraso, porque los motivos son los mismos: sólo puedo siquiera acercarme al QL los fines de semana, y en tres ocasiones seguidas ha sido imposible, por motivos de trabajo y personales. Y cuando ya estaba todo a punto, las unidades de disco fallan; son de 1987 las pobres, las que compré para el Spectrum con la interface Disciple, y ya empiezan a dar síntomas de demencia senil, así que hacer las copias ha sido como una peregrinación.

Para 1994 hay una propuesta de varios de entre nosotros, consistente en sacar sólo cuatro números y reducir la suscripción a 1500 pesetas. A mí me parece bien, porque como digo, estamos todos muy parados.

Yo sigo creyendo que la mejor solución sería hacernos un hueco en una BBS existente, solicitar un apartado sobre QDOS, de modo que sirviera para intercambiar textos, programas y mensajes continuamente entre todos. De este modo cada cual pagaría sólo el uso que hiciera del servicio, el intercambio sería mucho más frecuente, nadie cargaría con el muerto, etc. Las desventajas ya las conocemos también, así que no hace falta repetirlas. Este sistema podría combinarse con una pequeña hoja informativa mensual con las últimas novedades que pueden encontrarse en la BBS, para no ir a ciegas cada vez que uno se conecte y ahorrar teléfono.

El 13 de septiembre estuve con nuestro amigo Salvador Merino en Fuengirola, cambiando impresiones. En una última conversación me ha propuesto volver a tomar la redacción de QLíper, algo que ya habíamos hablado, pero que habíamos dejado pendiente hasta terminar el año. Ya me he convencido del todo de que no puedo sacar los números a tiempo, por más que lo he intentado, porque el trabajo me roba más y más tiempo y no tiene aires de cambiar, así que a partir de 1994 se hará cargo de todo otra vez Salvador Merino. Por otra parte, el tiempo que yo lograba dedicaba a QLíper podré dedicarlo a hacer colaboraciones, cosa que en los últimos meses ha sido imposible, de modo que mejor todavía, más material... ¡y puntual!, porque sabemos que Salvador podrá sacar cada número sin ningún retraso, ya que puede dedicar ratos libres a lo largo del día al QL.

Salvador: tal como hablamos, te mando en un paquete el material sobrante: los discos, sobres, el dinero de caja, la base de datos de socios, unas etiquetas ya impresas para mandar el QLíper 47 y la nota del coste de enviar este número. Mi cuota para el año que viene te la enviaré cuando sepamos cuál es, si 2000 o 1500 pesetas. Ah, por cierto, con este número envío una nota a todos los grupos con los que intercambiamos la revista para que a partir de ahora te envíen las suyas a tu dirección.

Que os lo paséis bien, etc., etc.

marcos, diciembre de 1993

MENSAJES
========

¿Alguien tiene noticia de Dasio Carballeira Tella? Su QLíper 45 vino devuelto, con la típica nota del cartero: "se ausentó"... Veremos qué pasa con el 46.

Nueva dirección de Rafael Illanes:

Rafael Illanes Muñoz
Salvador Dalí 4 esc. dcha. 1º-B
28933 MóSTOLES
teléfono: 91-6137088

Para Francisco Díaz-Tendero:

¿Pero qué haces con los discos, tío? Los números 44 y 45 me los devuelves como si les hubiera pasado por encima una manada de elefantes locos. Si te siguen llegando los discos estropeados, creo que lo mejor será enviártelos a tu dirección particular en vez de a la base naval. Sospecho que les hacen pasar por algún detector de explosivos a base de rayos kripton y martillazos o algo así, porque eres el único al que le llegan hechos polvo. En fin, te envío los números 44, 45 y 46 juntos, a ver si a la enésima va la vencida.

EN ESTE DISCO...
================

La descripción del directorio se ha hecho según el siguiente esquema:

```
- nombre_de_grupo [ORIGEN DEL PROGRAMA]
nombre_de_fichero (AUTOR): TEMA
nombre_de_fichero [COMENTARIO]
```

En un grupo de ficheros, el nombre del autor y el tema aparece sólo en el fichero de texto que aclara el contenido de los demás ficheros y su utilización.

```
QLíper_46_txt
powerpc_txt                   (Salvador Merino): procesadores RISC PowerPC
socios_QLíper_fxi
socios_QLíper_fxd
- TIL Olimpo
til_until_h
til_traqdos_c
til_olim1_c
til_olim2_c
til_until2_h
til_help_olp
til_olimpo_exe
til_demomenu_olp
til_boot_olp
til_demo_olp
til_demoazar_olp
til_demoprograma_olp
til_demoexplica_olp
til_demopantalla_olp
til_demotiempo_olp
til_democuadro_olp
til_democadena_olp
til_demotecla_olp
til_democonfig_olp
til_demopulsa_olp
til_demofichero_olp
til_demousuario_olp
olim2_c
olimpo_exe
demo_olp
democonfig_olp
demopulsa_olp
demofichero_olp
demousuario_olp
demo_cnf
Olimpo_txt                    (Pedro Reina): Descripción de Olimpo
Olimpo_h
libOlimpo_a
til_olimpo_txt                (Salvador Merino): TIL OLIMPO
- Calc
```

calc_app
help_app
calculadora_c
until2_h
calc_exe
traqdos_c
calc_txt (Salvador Merino): lenguaje escrito en C
- Inteligencia artificial
menú1_proc_bas
menú0_proc_bas
lisanjous_3_bas
itinerario_topográfico_bas
sistema_complejo_bas
leer_disco_bas
complejos_bas
tranferir_bas
inteligencia_artificial_1_txt (Rafael Illanes): varios
sistema_ecuaciones_bas
- Test de reflejos [programoteca de dominio público de SIN_QL_AIR]
React_Sb

LAS CUENTAS DE QLíper

Fecha	Concepto	ESP	Saldo
93.01.01	Saldo anterior		+6820
93.01.11	Suscripción de Javier ZUBIETA	+2000	+8820
93.01.12	Suscripción de Miguel ESTARELLAS	+2000	+10820
93.01.12	De Miguel Estarellas por envío anterior	+200	+11020
93.01.14	Suscripción de Josu REGIDOR	+2000	+13020
93.01.14	Suscripción de Francisco DIAZ-TENDERO	+2000	+15020
93.01.14	Suscripción de Pablo CARDENAS	+2000	+17020
93.01.14	De Francisco y Pablo por envío anterior	+500	+17520
93.01.14	Suscripción de Diego MORIARTY	+2000	+19520
93.01.14	Comisión robada por el Banco Santander	-300	+19220
93.02.04	11 sobres acolchados	-396	+18824
93.02.04	Suscripción de Mariano BERGES	+2000	+20824
93.02.04	Sellos para QLíper 42	-2006	+18818
93.02.22	Sellos para QLíper 42 a SIN_QL_AIR	-135	+18683
93.02.22	Suscripción de Pedro REINA	+2000	+20683
93.02.22	Contribución de Pedro REINA	+1000	+21863
93.03.01	Suscripción de José Carlos DE PRADA	+2000	+23863
93.03.02	Suscipción de Felipe BERGANZA	+2000	+25863
93.03.23	20 discos Verbatim	-1790	+24073
93.04	25 sobres acolchados	-900	+23173
93.05.04	Sellos para enviar el QLíper 43	-1648	+21525
93.05.15	Sellos para enviar 2 QLíper 43 más	-103	+21422
93.06.02	Suscripción de Ian-Charles COLEMAN	+2000	+23422
93.06.14	Sellos para 1 QL 42 a COLEMAN	-40	+23382
93.06.14	Sellos para 1 QL 43 en MS-DOS	-40	+23342
93.06	30 sobres acolchados	-1080	+22262
93.06	50 discos Verbatim	-4250	+18012
93.07.12	Franqueo de QLíper 44	-1419	+16593
93.08.22	10 sobres acolchados	-360	+16233
93.08.31	Suscripción de Joaquín GALLARDO	+2000	+18233
93.09.06	Franqueo de QLíper 45	-1428	+16085
93.09	Franqueo de carta a IQLR sobre QLíper	-90	+16995
93.09.05	Reventa de 30 sobres reciclados	+90	+16085
93.11.02	50 discos 3M Highland	-3250	+13555
93.11.29	30 sobres acolchados	-1200	+12355

Total actual en caja: .... +12355

Los discos de QLíper:

Fecha	Cantidad	Total
91.10.23	+50	50

92.01.07	-20	(QLíper 36)	30	
92.03.03	+50		80	
92.03.04	-20	(QLíper 37)	60	
92.04.25	-20	(QLíper 38)	40	
92.06.30	-20	(QLíper 39)	20	
92.08.24	+50		70	
92.09.09	-34	(QLíper 40)	36	
92.11.17	+50		86	
92.11.18	-34	(QLíper 41)	52	
93.02.04	-27	(QLíper 42)	25	
93.02.22	-1	(QLíper 42)	24	
93.03.24	+20		44	
93.05.04	-21	(QLíper 43)	23	
	-14	(QITALY 11)	9	
93.05.15	-2	(QLíper 43)	7	
	-2	(QITALY 11)	5	
93.06.13	-1	(QLíper 42)	4	
	-1	(QLíper 43 DOS)	3	
93.06	+50		53	
93.07	-16	(QLíper 44)	37	(suscriptores)
	-5	(QLíper 44)	32	(grupos extranjeros)
93.09.04	-17	(QLíper 45)	16	(suscriptores)
	-5	(QLíper 45)	11	(grupos extranjeros)
93.09.04	-3		8	(QLíperes 42, 43 y 44 a J. Gallardo)
93.11.02	+50		58	
93.11.14	-1	(estropeado)	57	
	-4	(Redacción)	53	

Discos disponibles para enviar: ..			53	

QLíper
Redactor: Salvador Merino
Tel. +34-(9)5-2475043
Cerámicas Mary
Ctra. Cádiz (Torreblanca del Sol)
ES-29640 FUENGIROLA

ULTIMAS NOTICIAS SOBRE LA FAMILIA DE MICROPROCESADORES RISC PowerPC

La revista BYTE de agosto'93 estaba dedicada al PowerPC. La familia comienza con el MPC601 (32 bits de direcciones y 64 bits de datos) y termina con el MPC620 (64 bits de direcciones y datos).

La versión de 66 MHz del PowerPC 601 tiene un precio inicial de $450, que es la mitad del precio del PENTIUM de Intel a 66 MHz.

El PowerPC requiere para funcionar solamente 9 W, y el Pentium 16 W. Al funcionar el PowerPC más fresco que el Pentium, necesita menos ventiladores, disipadores de calor y aire acondicionado.

La velocidad del PowerPC es de 1.5 (modo emulación) a 5 veces (modo nativo) la del Pentium.

Sistemas operativos disponibles para el PowerPC:

- Apple System 7.
- IBM OS/2
- IBM AIX.
- Sun Microsystems "SOLARIS".
- PowerOpen Enviroment (sistema nativo del PowerPC).
- Taligent Object-oriented PINK (otro sistema nativo).

Otros sistemas disponibles para el próximo año son:

- Windows NT.
- Novell NetWare.
- Unix System V.

El PowerPC 601 emula un Motorola 68040 a su velocidad nativa. Hasta la presente solamente emula al MAC, pero con tal potencial, nada puede impedir que emule toda la familia de ordenadores 680XX.

Si os habéis fijado, todos los sistemas operativos del PowerPC son multitarea. Elegir cuál de ellos cubre nuestras necesidades o gustos será una tarea difícil.

Yo he pedido una Miracle Systems QXL, y cuando la reciba voy a comprar un PC modesto (un 386 SX o DX), pues sé que se va a quedar anticuado muy rápidamente, conociendo la evolución actual del mercado. Pero viendo las características de la familia PowerPC, y aunque ya hay tarjetas conteniendo un PowerPC 601 y 16 MB DRAM para PC (cuesta $5000 con un compilador 'C'), creo que ya sé qué ordenador voy a utilizar a principios del nuevo siglo XXI, y espero que Tony Tebby no se retrase en transportar su sistema QDOS al PowerPC.

IBM, MOTOROLA y APPLE se han apuntado un tanto en la conquista del mercado. Su objetivo es colocar en el mercado un millón de ordenadores PowerPC durante 1994, y su precio en el mercado se cree que puede ser muy similar a un PC 486DX.

Dejando de lado el PowerPC, he de advertir que los constructores de hardware están actualmente lanzando masivamente prototipos de ordenadores RISC con similares características a las del PowerPC y el Pentium. Pero todo va de lo mismo:

- Obtener la mayor velocidad y prestaciones posible.
- Correr el mayor número de sistemas operativos estándar posibles.
- Aprovechar el hardware estandár barato que ya existe para PC, VME,...

En resumen, yo diría que la evolución de los ordenadores ha sufrido desde 1986 hasta hoy un pequeño parón en beneficio de Microsoft (e INTEL), pero todo vuelve a la normalidad a que estabamos acostumbrados antes de la invasión PC. Incluso yo diría que el futuro es muchísimo más bonito de lo que se podía esperar.

Salvador Merino, 17/8/1993

```
#####
#   #  #    ###  ##   ##  ###   ####
#   #  #     #   # # # #  #  #  #  #
#   #  #     #   #  #  #  ###   #  #
#####  ###  ###  #     #  #     ####
```

```
-----------------------------------------------------------------
Sistema de programación en C multiplataforma orientado al objeto
-----------------------------------------------------------------
Versión 1.0              Pedro Reina               J.15.7.1993
-----------------------------------------------------------------
```

Indice
======

5. Relación alfabética de funciones
6. Cómo hacer cambios
7. Historia de las versiones
8. Errores
9. Futuras mejoras

Presentación

El sistema Olimpo pretende facilitar la tarea del programador en C en varios aspectos:

* Aumentar la legibilidad del código.
* Acortar el tiempo de desarrollo de aplicaciones.
* Reutilizar código.
* Transportar programas entre plataformas.

Por todo ello, he decidido implementar el sistema orientado a objetos. Comienzo con objetos de bajo nivel, que se comunican directamente con el hardware y hay que reescribir cuando se intenta llevar el sistema a una plataforma distinta. Estos objetos permiten uniformizar el tratamiento de las funciones de bajo nivel y sobre ellos se puede construir objetos más abstractos que dan servicios más complejos. En un futuro, pretendo crear objetos de más alto nivel, que se encarguen ellos mismos de parte de la gestión del programa.

De momento Olimpo está disponible para el QL usando el compilador C68 (v3.05) y para PC usando Turbo C++ (v1.01) como compilador de C. Pretendo escribir una versión sobre UNIX, usando COHERENT.

El sistema consta de un fichero de cabecera, Olimpo.h, y un fichero de librería, Olimpo.lib (o libOlimpo_a, en el QL).

El sistema es de dominio público. Se puede distribuir libremente siempre y cuando se copie completo. También es posible que algún desarrollador desee entregar los fuentes de su programa, pero sin cargar el disquete con todo el sistema Olimpo. En ese caso, se puede incluir sólo el fichero de cabecera y el de librería, pero hay que hacer constar que se ha usado Olimpo, y cómo se puede obtener en su versión completa. En definitiva, mi deseo es que lo utilice quien quiera de modo que cada vez más gente conozca su existencia y lo pueda conseguir si lo desea.

Cualquier sugerencia, comentario o pregunta será bien recibido. Estos son mis datos:

Pedro Reina
c/ Sandoval 6
28010 Madrid
España

Teléfono: 91 - 593 81 59

Observación:
A lo largo de la documentación se utiliza como separador entre el nombre de un fichero y su extensión el carácter '.', el habitual en UNIX y MS-DOS. En el caso del QDOS, el separador que realmente se utiliza es '_'.

Ficheros del disquete

Están disponibles dos disquetes: uno para QL y otro para PC. La relación de los ficheros esenciales es la misma en los dos casos:

Olimpo.txt
Olimpo.h
Olimpo.lib   (En el QL, libOlimpo_a)

Demo.c
Demo.cnf
Demo.exe

Olimpo.txt contiene esta documentación.
Olimpo.h es el fichero de cabecera.
Olimpo.lib es el fichero de librería.
Demo.c es el código de un programa de demostración del sistema.
Demo.cnf es el fichero de configuración de Demo.
Demo.exe es el ejecutable que permite ver una pequeña demostración de las posibilidades de Olimpo.

El directorio Fuente contiene todos los ficheros fuente del sistema. Son idénticos para los dos ordenadores.

El directorio Valida contiene los ficheros que se utilizan para validar las funciones del sistema según se va desarrollando. También son idénticos para ambos ordenadores.

El directorio Uti contiene varias utilidades para crear el sistema, que dependen del ordenador.

Uso del sistema

Para utilizar el sistema hay que instalarlo primero. He intentado que esto sea lo más sencillo posible. De hecho, prácticamente no es necesaria instalación, ya que para compilar un programa con Olimpo basta incluir Olimpo.h y al invocar el montador, añadir Olimpo.lib. Para conseguir que esto se realice fácilmente, lo mejor es copiar Olimpo.h en el directorio de ficheros se cabecera del compilador (normalmente, INCLUDE) y Olimpo.lib en el de ficheros de librería (usualmente, LIB)

Para compilar un programa usando Olimpo:

1. Escribir antes del código:

```
#include <Olimpo.h>
```

2. Escribir como primera línea de código del main():

```
Pan_Define (PAN_TEXTO);
```

3. Escribir como última línea del main() [o como penúltima, caso de que el main() acabe con return()]:

```
Pan_Cierra();
```

4. Al montar, añadir Olimpo.lib.

Con Turbo C, se puede usar esta orden:

```
TCC Programa Olimpo.lib
```

Con C68, ésta:

```
EX CC; "Programa_c -lOlimpo"
```

Siguiendo estas instrucciones, el programa "Hola, mundo" se escribiría así en el sistema Olimpo:

```
#include <Olimpo.h>
main()
  {
  Pan_Define (PAN_TEXTO);
  Pan_PonTexto (10,30,"Hola, mundo");
  Usr_PulsaUnaTecla ("Programa concluido");
  Pan_Cierra ();
  return (0);
```

}

Para ejecutar los programas producidos con la ayuda de Olimpo:

En el PC:

Si se utiliza la pantalla en modo texto, no hay ninguna restricción.
Si se utiliza la pantalla en modo gráfico, el PC debe tener tarjeta VGA o compatible.

En el QL:

Si no se utiliza el entorno de puntero ("Pointer Environment") hay que arrancar los programas con EXEC_W o EW.
Si se dispone del entorno de puntero (PE), se puede arrancar también con EXEC o EX.
En cualquier caso, la pantalla debe estar en modo de alta resolución (MODE 4).

Descripción del sistema

El sistema está distribuido por objetos. Cada uno se ocupa de una parte de la programación, y he procurado hacerlos lo más independientes entre sí que podido.

Hay un fichero de definiciones generales, en la que se tiene:

La definición del ordenador objetivo, mediante la definición del macro OLIMPO_PC o del OLIMPO_QL. Esto se puede utilizar para escribir distintas versiones del código que lo exija y, mediante compilación condicional, mantener un único fuente.

El tipo de datos "logico", con sus valores SI y NO.

El tipo de datos "contador", con un rango mínimo de valores de -32768 a 32767 (en el QL el rango es mayor).

El tipo de datos "entero", con un rango de valores de -2147483648 a 2147483647.

La constante NULO, que es un alias de NULL (carácter nulo).

La constante NIL, que es el 0 (se utiliza en LISP como fin de lista, y en ese sentido lo voy a usar en Olimpo).

Las funciones Abs(x), Max(x,y) y Min(x,y).

Cada objeto tiene un identificador de tres letras y todas las funciones y macros de ese objeto comienzan con ese identificador.

A continuación describo cada objeto:

Pantalla (Pan)

Este objeto permite controlar la apariencia completa de la pantalla.

La función Pan_Define() permite iniciar los valores por defecto necesarios. En esta versión se utiliza un parámetro, que puede ser PAN_TEXTO o PAN_GRAFICO, para indicar al PC si se desea tener la pantalla en modo texto o en modo gráfico. Esto permite incluir gráficos en el PC. Este debe tener tarjeta VGA. Si no se dispone de ella, el sistema arranca en modo texto.

La función Pan_Cierra() se utiliza normalmente al final de la aplicación para restaurar todo y volver al sistema operativo.

Es perfectamente posible usar en una parte central del programa Pan_Cierra() y

luego otra vez Pan_Define() para cambiar de modo, aunque el contenido de la pantalla se perderá.

La pantalla es de 24 filas numeradas de 0 a 23 comenzando por arriba y 80 columnas numeradas de 0 a 79 empezando por la izquierda.

Se dispone de 4 colores, definidos con los macros BLANCO, NEGRO, ROJO y VERDE. Se manejan con las funciones Pan_Papel(), Pan_Tinta() y Pan_Color(). Estas funciones conviene utilizarlas antes de escribir algo, ya que los colores de papel y de tinta se cambian muy a menudo al llamar a otros objetos, de modo que no hay ninguna garantía de que permanezcan como los determine el programador.

La función Pan_Cursor() controla la posición del cursor.

La función Pan_CursorVisible() determina si el cursor se ve en la pantalla o no. En el PC en modo gráfico no está disponible el cursor.

Las funciones Pan_Texto(), Pan_Entero() y Pan_Car() permiten escribir en la pantalla texto (cadenas), enteros de tipo "contador" (no de tipo "entero") y caracteres.
Las funciones Pan_PonTexto(), Pan_PonEntero() y Pan_PonCar() colocan el cursor en la posición indicada y luego escriben el texto, entero o carácter.

Se puede activar o desactivar el modo de escritura resaltada con la función Pan_Resalta(). La situación normal es la de escritura resaltada desactivada, así que es importante conectarla cuando se desee usar e, inmediatamente, desconectarla. En el PC en modo gráfico no se dispone de esta posibilidad.

La función Pan_Limpia() borra la pantalla completa del color actual del papel.

La función Pan_BorraLinea() borra una linea con el color indicado.

La función Pan_Borra() borra una zona rectangular con el color indicado.

Azar (Azr)

Se pueden obtener números enteros aleatorios en el margen que se desee. Basta iniciar el generador al principio con Azr_Inicia() y luego ir pidiendo enteros con Azr_Entero().

Tiempo (Tim)

La función Tim_Crono() devuelve una referencia temporal de precisión de tipo "tiempo". Si se restan dos valores devueltos por Tim_Crono(), se obtiene el tiempo real transcurrido entre las dos llamadas.
La función Tim_Espera() obliga al programa a suspender su ejecución un periodo de tiempo determinado.

Cadena (Cad)

El manejo de cadenas en C no es muy cómodo. En esta implementación me he decidido por utilizar cadenas de asignación dinámica, por ser una aproximación más flexible que las cadenas de asignación estática.

He definido el tipo "cadena", que es simplemente un puntero a char; es decir, si declaramos una variable como tipo cadena, no se reserva automáticamente memoria para ella, sino que hay que pedirla explícitamente y luego liberarla. Muchas de las funciones definidas en este objeto y otros que lo utilizan realizan la reserva de memoria sin que el programador se tenga que ocupar.

Esto permite trabajar de dos formas distintas con las cadenas:

Por ser punteros, se pueden asignar a cadenas constantes. Por ejemplo, es válido:

```
cadena Cad;
Cad = "Esto es una cadena";
  /* Aquí tu código */
Cad = "Y esto es otra cadena";
```

Y por otro lado, se pueden crear cadenas de la longitud precisa en cada momento:

```
cadena Cad;
Cad = Cad_Crea (10);
Cad_Copia (Cad,"0123456789");
  /* Aquí tu código */
Cad_Destruye (Cad);
```

Las funciones más importantes son las que reservan memoria y la liberan.

Cad_Crea() reserva memoria para una cadena de la longitud indicada (reservando un octeto más para el carácter NULL), le da el valor inicial de cadena vacía y devuelve la dirección de memoria reservada (que es un puntero a char). La memoria así reservada no se libera automáticamente al salir de la función en la que se reserva, de modo que es responsabilidad del programador destruirla cuando no sea necesaria.

Cad_Destruye() es la encargada de liberar la memoria reservada mediante Cad_Crea(). También se debe usar para liberar memoria reservada por otras funciones.

Sonido (Son)

Este objeto se encarga de las señales acústicas. Se pueden conectar con Son_Enciende(), desconectar con Son_Apaga() o cambiar de estado con Son_Cambia().

Las llamadas a las funciones de este objeto no provocan inmediatamente la generación del sonido, eso dependerá de si en ese momento está conectado o no.

Tecla (Tec)

He definido el tipo "tecla" y la función Tec_Pulsada() como básicos para poder olvidar las grandes diferencias en el tratamiento de las teclas en los diversos sistemas operativos. En Olimpo sólo hay que saber que Tec_Pulsada() devuelve la tecla que haya pulsado el usuario, que pertenece al tipo "tecla". Junto con eso, hay toda una colección de macros que definen cada tecla; por ejemplo, la tecla F1 está definida con el macro TEC_F1, y el programador no necesita conocer qué codigo (o códigos) asigna el sistema operativo a F1 (cada uno lo hace de una forma distinta). En esta versión no dispongo de macros para la ñ ni las vocales acentuadas.

Se puede consultar en cualquier momento cuál fue la última tecla pulsada por el usuario con Tec_Ultima(), lo que resulta muy cómodo para obtener información en funciones muy alejadas de aquellas en las que el programa interactúa con el usuario.

Se puede pedir una tecla que pertenezca a un determinado rango con Tec_Validada() e incluso hacerlo con vuelta inmediata, si no hay tecla correcta disponible, con Tec_ValidadaRapido().

Hay funciones para pasar a mayúsculas o minúsculas, pero en esta versión sólo para caracteres ASCII (hasta 127).

No he definido macros para todas las teclas que son posibles en cada ordenador, sino sólo para aquellas que son perfectamente compatibles. En el PC existen algunas teclas que no están disponibles en el QL, pero con funciones que normalmente en el QL se asignan a determinadas combinaciones de tecla.

Esta es la lista de los macros que corresponden a distintas pulsaciones en PC y

en QL:

Macro	Tecla PC	Tecla QL
TEC_INICIO	Inicio	ALT/Cursor izquierda
TEC_FIN	Fin	ALT/Cursor derecha
TEC_AVPAG	AvPág	ALT/Cursor abajo
TEC_REPAG	RePág	ALT/Cursor arriba
TEC_CTRL_INICIO	CTRL/Inicio	CTRL/ALT/Cursor izquierda
TEC_CTRL_FIN	CTRL/Fin	CTRL/ALT/Cursor derecha
TEC_CTRL_AVPAG	CTRL/AvPág	CTRL/ALT/Cursor abajo
TEC_CTRL_REPAG	CTRL/RePág	CTRL/ALT/Cursor arriba
TEC_INSERTAR	Insert	MAY/CAPSLOCK
TEC_SUPRIMIR	Supr	CTRL/Cursor derecha
TEC_RETROCESO	Retroceso	CTRL/Cursor izquierda

Y esta es la de teclas comunes:

```
TEC_ESC  TEC_ENTER  TEC_ESPACIO
TEC_ARRIBA  TEC_ABAJO  TEC_IZQUIERDA  TEC_DERECHA
TEC_TAB  TEC_MAY_TAB
TEC_A ... TEC_Z
TEC_MAY_A ... TEC_MAY_Z
TEC_0  TEC_1 ... TEC_9
TEC_CTRL_A ... TEC_CTRL_Z
TEC_ALT_A ... TEC_ALT_Z
TEC_ALT_0  TEC_ALT_1 ... TEC_ALT_9
TEC_F1  TEC_F2  TEC_F3  TEC_F4  TEC_F5
TEC_MAY_F1  TEC_MAY_F2  TEC_MAY_F3  TEC_MAY_F4  TEC_MAY_F5
TEC_CTRL_F1  TEC_CTRL_F2  TEC_CTRL_F3  TEC_CTRL_F4
TEC_ALT_F1  TEC_ALT_F2  TEC_ALT_F3  TEC_ALT_F4  TEC_ALT_F5
```

Cuadro (Cdr)

Se pueden representar cuadros, cajas y líneas de cualquier color disponible y en dos grosores distintos.

El color se indica con Cdr_Color() y para representar los grosores de las líneas se tienen los macros CDR_SIMPLE y CDR_DOBLE.

Usuario (Usr)

Las dos últimas filas de la pantalla están reservadas para interactuar con el usuario. Hay funciones para darle indicaciones, informarle de acciones y también para que edite enteros y texto (cadenas).

Fichero (Fch)

La función Fch_AbreLeer() permite abrir un fichero para leer datos de él y Fch_AbreGrabar() abrirlo para grabar datos. Ambas cosas se pueden hacer un modo texto y en modo binario, y para indicarlo están los macros FCH_TEXTO y FCH_BINARIO. Naturalmente, esta distinción es necesaria por culpa del MS-DOS; de hecho, en el QL no hay ninguna diferencia entre los dos modos.

Una vez abierto un fichero, se pueden leer y escribir líneas con Fch_LeeLinea() y Fch_EscribeLinea() y grupos de octetos con Fch_LeeOcteto() y Fch_EscribeOcteto()

Las funciones se encargan de gestionar los errores que se puedan producir. Por ejemplo, al llamar a Fch_Borra(), primero se pide confirmación al usuario, luego se comprueba la existencia del fichero y por último se emite un mensaje indicando si ha sido posible el borrado o no.

Menú (Men)

Hay menús horizontales y verticales. Sólo hay que preocuparse de asignar la zona de la pantalla en la que se desea que aparezca el menú, el objeto se ocupa de la colocación. Las opciones pueden llevar una tecla caliente, un atajo o "hotkey", que permite elegir la opción simplemente pulsando esa tecla. Para indicar cuál es, se utiliza el carácter '>' antes de la letra en la llamada al objeto.

Programa (Prg)

Este objeto se ocupa simplemente de presentar información sobre el programa en la línea superior de la pantalla. Si no se utiliza, se dispone de una línea más para el desarrollo del programa.

Configuración (Cnf)

Este objeto permite leer información de un fichero filtrando todos los comentarios que se hayan escrito en él. El programador debe decidir después cómo lee la información entregada por el objeto, que lo hace en forma de cadena.

En los ficheros de configuración se considera línea de comentario toda aquella que tenga un '*' (asterisco) como primer carácter, aunque haya caracteres en blanco antes de él.

También se pueden poner comentarios dentro de una línea poniendo "//" (doble barra). Todo lo que haya a continuación, se considera comentario.

Para ampliar información sobre el uso de los objetos, consulta la relación alfabética de funciones y mira el fichero de demostración (Demo.c) y los de validación de funciones (V*.c)

Relación alfabética de funciones
================================

```
FUNCION:  Abs()
OBJETIVO: Calcular el valor absoluto de un número
ENTRADAS: Un número
SALIDAS:  El valor absoluto
EJEMPLO:  Abs(-3)

FUNCION:  Azr_Entero ()
OBJETIVO: Obtener un número entero aleatorio en un rango especificado
ENTRADAS: a: número más bajo admitido; b: el más alto admitido
SALIDAS:  un número entero entre a y b (ambos inclusive)
EJEMPLOS: Azr_Entero (0,100)
          Azr_Entero (-10,10)

FUNCION:  Azr_Inicia ()
OBJETIVO: Iniciar el generador con un número aleatorio
ENTRADAS: Ninguna, se lee el reloj del sistema
SALIDAS:  Ninguna
EJEMPLO:  Azr_Inicia ()

FUNCION:  Cad_Cambia()
OBJETIVO: Cambiar en una cadena un carácter por otro
ENTRADAS: Cadena destino, carácter viejo, carácter nuevo
SALIDAS:  La cadena queda modificada y se devuelve
EJEMPLO:  Cad_Cambia (Cadena,'a','b')

FUNCION:  Cad_CarPertenece ()
OBJETIVO: Averiguar si un caracter pertenece a una cadena
```

```
      ENTRADAS: La cadena y el caracter
      SALIDA:   La posición que ocupa el carácter en la cadena, o 0
      NOTA:     Se empieza a contar en 1
      EJEMPLO:  Cad_CarPertenece (Nombre,'a')

      FUNCION:  Cad_Copia()
      OBJETIVO: Copiar una cadena en otra
      ENTRADAS: La cadena destino y la cadena origen
      SALIDAS:  La cadena destino
      EJEMPLO:  Cad_Copia (Nombre,"Eustaquio")

      FUNCION:  Cad_Crea()
      OBJETIVO: Reservar memoria para almacenar una cadena
      ENTRADAS: El número máximo de caracteres de la cadena
      SALIDAS:  Una cadena
      NOTAS:    Se reserva espacio automáticamente para el NULO
                La cadena queda iniciada a cadena vacía
                La cadena devuelta hay que destruirla cuando no sea necesaria
      EJEMPLO:  Cad_Crea (80)

      FUNCION:  Cad_Destruye()
      OBJETIVO: Liberar la memoria reservada para una cadena
      ENTRADAS: La cadena
      SALIDAS:  Ninguna
      EJEMPLO:  Cad_Destruye (Cadena)

      FUNCION:  Cad_Entero()
      OBJETIVO: Convertir un número entero en una cadena
      ENTRADAS: El número
      SALIDAS:  La cadena
      NOTA:     La cadena devuelta hay que destruirla cuando no sea necesaria
      EJEMPLO:  Cad_Entero (130)

      FUNCION:  Cad_Longitud()
      OBJETIVO: Calcular la longitud de una cadena
      ENTRADAS: Una cadena
      SALIDAS:  Su longitud
      EJEMPLO:  Cad_Longitud (Nombre)

      FUNCION:  Cad_Mueve()
      OBJETIVO: Mover hacia la derecha todos los caracteres de una cadena
                desde una posición determinada, cuidando que la longitud
                no exceda un cierto máximo
      ENTRADAS: La cadena, la posición y el máximo
      NOTA:     Las posiciones se cuentan a partir de cero
      SALIDAS:  La cadena queda modificada y se devuelve
      EJEMPLO:  Cad_Mueve (Cadena,2,40)

      FUNCION:  Cad_PrimerUtil (Cadena)
      OBJETIVO: Decir la posición del primer carácter de una cadena que no
                sea ni blanco, ni tabulador, ni retorno de carro ni nueva línea
      ENTRADAS: La cadena
      SALIDA:   La posición que ocupa el primer carácter útil, o 0
      NOTA:     Se empieza a contar en 1
      EJEMPLO:  Cad_PrimerUtil ( "   Ahora" )

      FUNCION:  Cad_QuitaCar()
      OBJETIVO: Quitar el caracter que ocupa una determinada posición
                en una cadena
      ENTRADAS: La cadena y la posición que hay que suprimir
      NOTA:     Las posiciones se cuentan a partir de cero
      SALIDAS:  La cadena queda modificada y se devuelve
      EJEMPLO:  Cad_QuitaCaracter (Cadena,2)

      FUNCION:  Cad_Subcadena (Larga,Corta)
      OBJETIVO: Averiguar si la cadena Corta está contenida en la Larga
      ENTRADAS: Las dos cadenas
      SALIDA:   La posición que ocupa el primer carácter de Larga donde
                comienza la primera aparición de Corta, o 0
      NOTA:     Se empieza a contar en 1
```

```
EJEMPLO:  Cad_Subcadena ( "Calamares","mar" )

FUNCION:  Cad_Trozo()
OBJETIVO: Extraer una cadena de otra
ENTRADAS: Cadena origen y puntos de comienzo y fin del corte
SALIDAS:  La cadena destino
NOTAS:    El primer carácter lleva el número 1
          La cadena devuelta hay que destruirla cuando no sea necesaria
EJEMPLO:  Cad_Trozo (Nombre,3,7)

FUNCION:  Cad_Une()
OBJETIVO: Unir varias cadenas
ENTRADAS: Las cadenas que hay que unir, terminadas por NIL
SALIDAS:  Una cadena conteniendo la unión
NOTA:     La cadena devuelta hay que destruirla cuando no sea necesaria
EJEMPLO:  Cad_Une ("Veinte de ", Mes, " de 1808",NIL)

FUNCION:  Cdr_Caja()
OBJETIVO: Dibujar una caja
ENTRADAS: Tipo de la caja y sus coordenadas
SALIDAS:  Ninguna
EJEMPLO:  Cdr_Caja (CDR_DOBLE,1,1,10,70)

FUNCION:  Cdr_Color()
OBJETIVO: Establecer el color con el que se dibujarán todas las figuras
ENTRADAS: El color
SALIDAS:  Ninguna
EJEMPLO:  Cdr_Color ( ROJO )

FUNCION:  Cdr_Dibuja()
OBJETIVO: Dibujar un cuadro
ENTRADAS: El tipo de línea que se dibuja en las líneas externas,
          el de las internas, el número de filas, el alto de cada
          una, el número de columnas, el ancho de cada una, la fila
          y columna de la esquina superior izquierda
SALIDAS:  Ninguna
EJEMPLO:  Cdr_Dibuja (CDR_DOBLE,CDR_SIMPLE,2,3,4,1,3,3)

FUNCION:  Cdr_Linea()
OBJETIVO: Dibujar una línea
ENTRADAS: El tipo de línea y las coordenadas
SALIDAS:  Ninguna
EJEMPLO:  Cdr_Linea (CDR_DOBLE,2,3,2,71)

FUNCION:  Cnf_Abre()
OBJETIVO: Abrir un fichero de configuración
ENTRADAS: El nombre del fichero
SALIDAS:  El fichero abierto o NULO si ha habido algún error
EJEMPLO:  Cnf_Abre ( "Datos.dat" )

FUNCION:  Cnf_Cierra ()
OBJETIVO: Cerrar un fichero de configuración
ENTRADAS: El fichero
SALIDAS:  NULO si todo va bien, EOF si hay error
EJEMPLO:  Cnf_Cierra ( FicheroConfig )

FUNCION:  Cnf_Lee()
OBJETIVO: Obtener la siguiente línea de un fichero de configuración
ENTRADAS: El fichero
SALIDAS:  La cadena si no hay error, o NIL
NOTAS:    La cadena no contiene '\n'
          La cadena devuelta hay que destruirla cuando no sea necesaria
EJEMPLO:  Cnf_Lee ( Entrada )

FUNCION:  Fch_Abre()
OBJETIVO: Abrir un fichero
ENTRADAS: El nombre del fichero y el tipo
SALIDAS:  El fichero abierto o NULO si ha habido algún error
EJEMPLO:  Fch_Abre ( "Datos.dat", "r" )
```

```
FUNCION:  Fch_AbreGrabar()
OBJETIVO: Abrir un fichero para grabar en él
ENTRADAS: El nombre del fichero y el modo: texto o binario
SALIDAS:  El fichero abierto o NULO si ha habido algún error o negativa
EJEMPLO:  Fch_AbreGrabar ( "Datos.bin",FCH_BINARIO )

FUNCION:  Fch_AbreLeer()
OBJETIVO: Abrir un fichero para leerlo
ENTRADAS: El nombre del fichero y el modo: texto o binario
SALIDAS:  El fichero abierto o NULO si ha habido algún error
EJEMPLO:  Fch_AbreLeer ( "Datos.dat",FCH_TEXTO )

FUNCION:  Fch_Borra()
OBJETIVO: Borrar un fichero
ENTRADAS: El nombre del fichero
SALIDAS:  Ninguna
EJEMPLO:  Fch_Borra ( "Viejo.txt" )

FUNCION:  Fch_Cierra ()
OBJETIVO: Cerrar un fichero
ENTRADAS: El fichero
SALIDAS:  NULO si todo va bien, EOF si hay error
EJEMPLO:  Fch_Cierra ( Salida )

FUNCION:  Fch_EscribeLinea()
OBJETIVO: Escribir una línea en un fichero (es decir, se pone '\n')
ENTRADAS: El fichero y una cadena
SALIDAS:  El fichero si no hay error, o NIL
EJEMPLO:  Fch_EscribeLinea ( Salida, Linea )

FUNCION:  Fch_EscribeOcteto()
OBJETIVO: Escribir en un fichero cierta cantidad de octetos
ENTRADAS: El fichero, la zona de memoria de donde se toman los octetos
          y la cantidad de octetos
SALIDAS:  El fichero o NIL si ha habido algún error
EJEMPLO:  Fch_EscribeOcteto ( Salida, Item, 1024 )

FUNCION:  Fch_Existe()
OBJETIVO: Informar si un fichero existe
ENTRADAS: El nombre del fichero
SALIDAS:  Lógica
EJEMPLO:  Fch_Existe ( "Datos.dat" )

FUNCION:  Fch_LeeLinea()
OBJETIVO: Obtener la siguiente línea de un fichero
ENTRADAS: El fichero
SALIDAS:  La cadena si no hay error, o NIL
NOTAS:    La cadena no contiene '\n'
          La cadena devuelta hay que destruirla cuando no sea necesaria
EJEMPLO:  Fch_LeeLinea ( Entrada )

FUNCION:  Fch_LeeOcteto()
OBJETIVO: Leer de un fichero cierta cantidad de octetos
ENTRADAS: El fichero, la zona de memoria donde dejar lo leido y la
          cantidad de octetos
SALIDAS:  El fichero o NIL si ha habido algún error
EJEMPLO:  Fch_LeeOcteto ( Datos, Info, 1024 )

FUNCION:  Fch_Nombre()
OBJETIVO: Formar el nombre completo de un fichero uniendo el nombre,
          el separador y la extensión
ENTRADAS: El nombre y la extensión
SALIDAS:  La cadena con el nombre completo
NOTA:     La cadena devuelta hay que destruirla cuando no sea necesaria
EJEMPLO:  Fch_Nombre ( "Datos", "txt" )

FUNCION:  Max()
OBJETIVO: Calcular el máximo de dos números
ENTRADAS: Los números
SALIDAS:  El mayor de ellos
```

```
      EJEMPLO:  Max (2,5)

      FUNCION:  Men_Horizontal()
      OBJETIVO: Presentar al usuario un menú horizontal para que pueda
                elegir entre varias opciones, mediante las teclas del
                cursor o mediante un atajo (tecla caliente)
      ENTRADAS: La fila en la que se muestra el menú, las columnas entre
                las que se muestra, un vector de cadenas, terminado en
                NIL, en el que se señalan los atajos y la opción que hay
                resaltar en primer lugar
      SALIDAS:  Un número indicando la opción elegida, empezando a contar
                en 1, o 0 si no se elige ninguna
      EJEMPLO:  Men_Horizontal (1,1,78,{">Fichero","A>yuda",NIL},2)

      FUNCION:  Men_Vertical()
      OBJETIVO: Presentar al usuario un menú vertical para que pueda
                elegir entre varias opciones, mediante las teclas del
                cursor o mediante un atajo (tecla caliente)
      ENTRADAS: Las coordenadas de la esquina superior izquierda e inferior
                derecha de la zona asignada al menú, un vector de cadenas,
                terminado en NIL, en el que se señalan los atajos y la opción que
                hay que resaltar en primer lugar
      SALIDAS:  Un número indicando la opción elegida, empezando a contar
                en 1, o 0 si no se elige ninguna
      EJEMPLO:  Men_Vertical (1,10,8,20,{">Fichero","A>yuda",NIL},2)

      FUNCION:  Min()
      OBJETIVO: Calcular el mínimo de dos números
      ENTRADAS: Los números
      SALIDAS:  El menor de ellos
      EJEMPLO:  Min (2,5)

      FUNCION:  Pan_Borra()
      OBJETIVO: Borrar una zona de la pantalla
      ENTRADAS: El color y las coordenadas
      SALIDAS:  Ninguna
      EJEMPLO:  Pan_Borra ( ROJO,3,3,15,15 )

      FUNCION:  Pan_BorraLinea()
      OBJETIVO: Limpiar una línea determinada
      ENTRADAS: La línea
      SALIDAS:  Ninguna
      EJEMPLO:  Pan_BorraLinea ( 0 )

      FUNCION:  Pan_Caracter()
      OBJETIVO: Escribir un carácter en la pantalla
      ENTRADAS: El carácter
      SALIDAS:  Ninguna
      EJEMPLO:  Pan_Caracter ('A')

      FUNCION:  Pan_Cierra ()
      OBJETIVO: Dejar la pantalla preparada para volver al sistema operativo
      ENTRADAS: Ninguna
      SALIDAS:  Ninguna
      EJEMPLO:  Pan_Cierra ()

      FUNCION:  Pan_Color ()
      OBJETIVO: Cambiar el color del papel y la tinta
      ENTRADAS: El color del papel y de la tinta
      SALIDAS:  Ninguna
      EJEMPLO:  Pan_Color ( VERDE, NEGRO )

      FUNCION:  Pan_Cursor()
      OBJETIVO: Colocar el cursor en una posición determinada
      ENTRADAS: La fila y la columna
      SALIDAS:  Ninguna
      EJEMPLO:  Pan_Cursor ( 0,0 )

      FUNCION:  Pan_CursorVisible()
      OBJETIVO: Poner y quitar el cursor
```

```
ENTRADAS: Lógica
SALIDAS:  Ninguna
EJEMPLO:  Pan_CursorVisible ( SI )

FUNCION:  Pan_Define ()
OBJETIVO: Definir la pantalla principal del programa
ENTRADAS: El modo que se va a usar: texto o gráfico
NOTA:     Se usa una pantalla de 24 filas (de 0 a 23) y
          80 columnas (de 0 a 79)
EJEMPLO:  Pan_Define ( PAN_TEXTO )

FUNCION:  Pan_Entero()
OBJETIVO: Escribir un número entero en la pantalla en un ancho determinado
ENTRADAS: El número y el ancho
SALIDAS:  Ninguna
EJEMPLO:  Pan_Entero ( 12345,6 )

FUNCION:  Pan_Limpia ()
OBJETIVO: Borrar la pantalla completa
ENTRADAS: Ninguna
SALIDAS:  Ninguna
EJEMPLO:  Pan_Limpia()

FUNCION:  Pan_Papel ()
OBJETIVO: Cambiar el color de la pantalla
ENTRADAS: El nuevo color
SALIDAS:  Ninguna
EJEMPLO:  Pan_Papel ( VERDE )

FUNCION:  Pan_PonCar()
OBJETIVO: Escribir un carácter en cierta posición
ENTRADAS: La fila, la columna y el carácter
SALIDAS:  Ninguna
EJEMPLO:  Pan_PonCar (3,5,'A')

FUNCION:  Pan_PonEntero()
OBJETIVO: Escribir un entero en cierta posición
ENTRADAS: La fila, la columna, el entero y el ancho disponible
SALIDAS:  Ninguna
EJEMPLO:  Pan_PonEntero (3,5,12345,6)

FUNCION:  Pan_PonTexto()
OBJETIVO: Escribir un texto en cierta posición
ENTRADAS: La fila, la columna y el texto
SALIDAS:  Ninguna
EJEMPLO:  Pan_PonTexto (3,5,"Hola")

FUNCION:  Pan_Resalta()
OBJETIVO: Poner y quitar la situación de resaltado de caracteres
ENTRADAS: Lógica
SALIDAS:  Ninguna
EJEMPLO:  Pan_Resalta ( SI )

FUNCION:  Pan_Texto()
OBJETIVO: Escribir un texto en la pantalla
ENTRADAS: Una cadena
SALIDAS:  Ninguna
EJEMPLO:  Pan_Texto ("Hola, usuario")

FUNCION:  Pan_Tinta ()
OBJETIVO: Cambiar el color con el que se escribe en la pantalla
ENTRADAS: El nuevo color
SALIDAS:  Ninguna
EJEMPLO:  Pan_Tinta ( ROJO )

FUNCION:  Prg_Presenta()
OBJETIVO: Presentar el programa
ENTRADAS: Nombre, versión, autor y fecha
SALIDAS:  Ninguna
EJEMPLO:  Prg_Presenta ( "Programa","0.0","Pedro Reina","1993" )
```

```
FUNCION:  Son_Apaga()
OBJETIVO: Apagar el sonido
ENTRADAS: Ninguna
SALIDAS:  La variable Son_Encendido se modifica y se devuelve
EJEMPLO:  Son_Apaga()

FUNCION:  Son_Bien()
OBJETIVO: Señalar brevemente que algo es correcto
ENTRADAS: Ninguna
SALIDAS:  El valor de Son_Encendido
EJEMPLO:  Son_Bien()

FUNCION:  Son_Cambia()
OBJETIVO: Cambiar el estado del sonido (Encendido / Apagado)
ENTRADAS: Ninguna
SALIDAS:  La variable Son_Encendido se modifica y se devuelve
EJEMPLO:  Son_Cambia()

FUNCION:  Son_Enciende()
OBJETIVO: Encender el sonido
ENTRADAS: Ninguna
SALIDAS:  La variable Son_Encendido se modifica y se devuelve
EJEMPLO:  Son_Enciende()

FUNCION:  Son_Error()
OBJETIVO: Señalar un error
ENTRADAS: Ninguna
SALIDAS:  El valor de Son_Encendido
EJEMPLO:  Son_Error()

FUNCION:  Son_MalaTecla()
OBJETIVO: Señalar que la tecla pulsada no se admite
ENTRADAS: Ninguna
SALIDAS:  El valor de Son_Encendido
EJEMPLO:  Son_MalaTecla()

FUNCION:  Tec_Disponible()
OBJETIVO: Decir si está disponible alguna tecla
ENTRADAS: Ninguna
SALIDAS:  0 si no hay tecla disponible, 1 si la hay
EJEMPLO:  Tec_Disponible()

FUNCION:  Tec_FijadoMayus()
OBJETIVO: Decir si está activado el sujeta-mayúsculas
ENTRADAS: Ninguna, se lee del sistema
SALIDAS:  Lógica
EJEMPLO:  Tec_FijadoMayus()

FUNCION:  Tec_Mayus()
OBJETIVO: Convertir una tecla en mayúscula
ENTRADAS: Una tecla
SALIDAS:  La tecla convertida en mayúscula
NOTA:     En esta versión no se tienen en cuenta caracteres acentuados
EJEMPLO:  Tec_Mayus ( TEC_A )

FUNCION:  Tec_Minus()
OBJETIVO: Convertir una tecla en minúscula
ENTRADAS: Una tecla
SALIDAS:  La tecla convertida en minúscula
NOTA:     En esta versión no se tienen en cuenta caracteres acentuados
EJEMPLO:  Tec_Minus ( TEC_MAY_A )

FUNCION:  Tec_Pertenece()
OBJETIVO: Decidir si una tecla pertenece a cierto rango
ENTRADAS: Una tecla y un vector con las teclas admitidas, terminado en NIL
SALIDAS:  La tecla si pertenece o NIL si no pertenece
EJEMPLO:  Tec_Pertenece ( TEC_F1, {TEC_ESC,NIL} )

FUNCION:  Tec_Pulsada()
```

```
      OBJETIVO: Esperar a que el usuario pulse una tecla y devolver
                su código
      ENTRADAS: Ninguna
      SALIDAS:  El código asignado a la tecla pulsada
                La variable global Tec_Ultima_ queda modificada
      NOTA:     Deben usarse siempre los macros, puesto que los códigos
                dependen del compilador
      EJEMPLO:  Tec_Pulsada()

      FUNCION:  Tec_Ultima()
      OBJETIVO: Devolver la última tecla pulsada por el usuario
      ENTRADAS: Ninguna, se usa la variable global Tec_Ultima_
      SALIDAS:  El código asignado a Tec_Ultima_
      EJEMPLO:  Tec_Ultima()

      FUNCION:  Tec_Validada()
      OBJETIVO: Devolver una tecla de un determinado rango
      ENTRADAS: Un vector con las teclas admitidas, terminado en NIL
      SALIDAS:  La tecla pulsada
      EJEMPLO:  Tec_Validada ( {TEC_ESC,NIL} )

      FUNCION:  Tec_ValidadaRapido()
      OBJETIVO: Devolver una tecla de un determinado rango o NIL si no hay
                disponible ninguna tecla correcta
      ENTRADAS: Un vector con las teclas admitidas, terminado en NIL
      SALIDAS:  La tecla pulsada o NIL
      EJEMPLO:  Tec_ValidadaRapido ( {TEC_ESC,NIL} )

      FUNCION:  Tim_Crono()
      OBJETIVO: Obtener una referencia temporal de precisión
      ENTRADAS: Ninguna, se consulta el reloj del sistema
      SALIDAS:  Un número real indicando el número de segundos desde la
                medianoche

      FUNCION:  Tim_Espera()
      OBJETIVO: Suspender la ejecución del programa durante cierto tiempo
      ENTRADAS: El tiempo en segundos
      SALIDAS:  Ninguna

      FUNCION:  Usr_Avisa()
      OBJETIVO: Avisar al usuario de algo
      ENTRADAS: El texto del aviso
      SALIDAS:  Ninguna
      EJEMPLO:  Usr_Avisa ( "Operación incorrecta" )

      FUNCION:  Usr_BorraZona()
      OBJETIVO: Borrar la zona de interacción con el usuario
      ENTRADAS: El nuevo color de la zona
      SALIDAS:  Ninguna
      EJEMPLO:  Usr_BorraZona ( NEGRO )

      FUNCION:  Usr_Consulta()
      OBJETIVO: Preguntar algo al usuario
      ENTRADAS: El texto de la pregunta
      SALIDAS:  Lógica, según la contestación del usuario
      EJEMPLO:  Usr_Consulta ( "¿Quieres seguir?" )

      FUNCION:  Usr_Entero()
      OBJETIVO: Devolver un número determinado por el usuario después de
                editar el que se le ofrece
      ENTRADAS: El número que hay que editar, el ancho asignado, los valores
                mínimo y máximo admitidos, las coordenadas donde se edita y
                los colores
      SALIDAS:  El entero editado
      EJEMPLO:  Usr_Entero (100,4,-900,2000,1,1,BLANCO,NEGRO)

      FUNCION:  Usr_Indica()
      OBJETIVO: Mandar un mensaje al usuario
      ENTRADAS: Dos cadenas
      SALIDAS:  Ninguna
```

```
EJEMPLO:  Usr_Indica ( "Elige una opción", "Pulsa ENTER" )

FUNCION:  Usr_Informa()
OBJETIVO: Mandar un mensaje al usuario
ENTRADAS: El texto del mensaje
SALIDAS:  Ninguna
EJEMPLO:  Usr_Informa ( "Calculando" )

FUNCION:  Usr_PulsaUnaTecla()
OBJETIVO: Mandar un mensaje al usuario y esperar que pulse una tecla
ENTRADAS: El texto del aviso
SALIDAS:  Ninguna
EJEMPLO:  Usr_PulsaUnaTecla ( "Operación terminada." )

FUNCION:  Usr_Texto()
OBJETIVO: Devolver una cadena determinada por el usuario después de
          editar la que se le ofrece
ENTRADAS: La cadena que hay que editar, el ancho asignado,
          las coordenadas donde se edita y los colores
SALIDAS:  Una cadena
NOTA:     La cadena devuelta hay que destruirla cuando no sea necesaria
EJEMPLO:  Usr_Texto ("Artesonado",20,1,1,BLANCO,NEGRO)
```

Cómo hacer cambios

El código está dividido en la mayor cantidad posible de ficheros para facilitar la realización de cambios.

Cada objeto (y las definiciones generales) tiene un fichero de cabecera.

Los nombres de los ficheros *.c reflejan el objeto al que se refieren (las tres primeras letras) y la función más importante que definen.

Es muy importante validar las funciones que se cambien o añadan, y para ello se dispone de ficheros de validación, que tienen como nombre Vobj.c, donde obj son las tres letras que identifican cada objeto. El fichero de definiciones generales se identifica con "gen".

Cuando se realiza un cambio en un objeto, hay que comprobar si ese cambio afecta a otros objetos que dependan de él, por lo que es muy aconsejable seguir cierto orden al realizar cambios. Hay que ir de más básicos a más complejos.

Esta es la relación de interdependencias estre objetos:

Objetos necesarios	Objeto	Objetos que dependen de éste
	Pan	Cdr Usr Men Prg
	Azr	
	Tim	
	Cad	Usr Fch Men Prg
	Son	Tec Usr
Son	Tec	Usr Men
Pan	Cdr	Usr
Pan Cad Son Tec Cdr	Usr	Fch Men
Usr Cad	Fch	Cnf
Cad Pan Usr Tec	Men	
Pan Cad	Prg	
Fch	Cnf	

Una vez realizados los cambios, hay que volver a formar los ficheros Olimpo.h y Olimpo.lib.

Para crear Olimpo.h he escrito el programa UneH, que lee todos los ficheros de cabecera, les quita la información que no es necesaria, y escribe Olimpo.h.

Para crear Olimpo.lib se dispone del fichero CreaLib.bat.

En el PC hay una serie de Makefiles, llamados Vobj.mak (donde obj se sustituye por el identificador de cada objeto) que sirven para tener al día los ficheros de validación.

En el QL no me funciona la utilidad make, de modo que he optado por crear un programa SuperBasic de soporte del desarrollo. Precisamente se llama Soporte_bas.

Historia de las versiones

La idea de realizar Olimpo proviene de intentar resolver varios problemas:

Llevaba mucho tiempo programando y me encontraba que estaba escribiendo las mismas rutinas una y otra vez. Había que encontrar el método para utilizar las rutinas ya escritas en programas nuevos.

Tengo tres sistemas operativos y me gustaría que mis programas funcionaran, sin cambiar el código, en los tres. El lenguaje que me permite hacer esto es el C.

Me gustaría escribir métodos generales de resolución de problemas para estudiar inteligencia artificial, pero para poder presentar un programa debe tener un mínimo interfaz; por tanto, hay que empezar por programar las cosas básicas y sobre ellas ir montando métodos mas complejos.

Cuando se realiza cualquier sistema de programación es muy aconsejable que las nuevas versiones no cambien los protocolos de las funciones, para no tener que retocar el código de un programa escrito sobre el sistema al cambiar de versión. Desafortunadamente, yo no tengo claras ciertas cosas y además estoy escribiendo el sistema partiendo de algo sencillo y añadiéndole utilidades poco a poco; por tanto, será inevitable que algunas funciones cambien de protocolo conforme aumente el número de versión de Olimpo. Por supuesto, iré documentando los cambios.

Versión 0.0

Es la versión en la que se comienzan a construir las bases del sistema.

Se dispone de un conjunto de definiciones generales y de 12 objetos:

General, Pantalla, Azar, Tiempo, Cadena, Sonido, Tecla, Usuario, Fichero, Menú, Programa, Cuadro y Configuración.

El sistema funciona por medio del fichero Olimpo.h, cuyo único cometido es llamar a los distintos .h de los objetos. Por tanto, el tiempo de compilación es muy elevado.

Versión 0.1

Reúno todos los .h para formar el fichero Olimpo.h.

Retoco el objeto Fichero, añadiendo Fch_Borra() y un manejador de interrupciones en el PC.

Versión 1.0

Separo lo más posible el código en varios ficheros, que compilo por separado y creo un fichero de librería. Además, escribo un programa que permite unir todos los .h del sistema quitando todas las redundancias y formando Olimpo.h, único fichero de cabecera que debe utilizar el programador usuario del sistema.

Reescribo completamente el objeto Cadena, utilizando asignación dinámica.

El objeto Pantalla en el PC ahora admite trabajar en modo gráfico, lo que abre las posibilidades de crear y añadir objetos gráficos a Olimpo en un futuro.

Reescribo completamente los programas de validación y creo métodos para el mantenimiento de los ficheros.

Todas estan funciones han sido retocadas de modo que hay que revisar el código que las utilice:

Pan_Define(), Pan_Entero(), Usr_Consulta(), Usr_Texto(), Fch_Abre(), Fch_AbreLeer(), Fch_AbreGrabar(), Fch_LeeLinea(), Fch_EscribeLinea(), Fch_Nombre() y todas las del objeto Cadena.

Errores

Versión 0.0

Usr_Consulta() -> En la versión PC el cursor a veces aparece y a veces no. No he encontrado el patrón del error; sospecho que es problema de la BIOS.

Cad_ConvierteEntero() -> tiene distinta implementación en QL que en PC y no estoy seguro de que funcione bien para enteros muy grandes.

Objeto Fichero -> en la versión PC hay que añadir un manejo de interrupciones, usando harderr(), ya que las interrupciones de hardware, como intentar escribir un un disquete protegido de escritura, las trata el sistema y rompe el diseño de la pantalla. Añadido en la versión 0.1.

Versión 1.0

Usr_Consulta() -> aun habiendo cambiado la función Pan_CursorVisible(), el error presente en la versión 0.0 sigue apareciendo.

Futuras mejoras

Objeto Cuadro -> Deseo añadir Cdr_Construye(), que devuelva cadenas rellenas con cuadros, para poder mandarlos a la impresora.

Objeto Impresora -> hacer.

Objeto Ventana -> hacer. Se encargará exclusivamente de realizar gráficos.

Objeto Base de Datos -> Deseo añadir el manejo de ficheros .dbf, por ser un formato universalmente aceptado, lo que permite la compatibilidad con gran cantidad de herramientas. Más adelante intentaré introducir ficheros de índice.

Objeto Lista -> hacer. Este objeto permitirá quitar ciertas restricciones en el manejo de algunos objetos, que ahora usan asignación estática. En particular, deseo escribir una rutina que permita elegir un fichero desde una lista de ficheros disponibles.

TIL OLIMPO, El lenguaje de QLíper para QLíper

El interprete/compilador OLIMPO v1.0 está basado principalmente en tres cosas:

- Un THREADED INTERPRETER LANGUAGE (TIL). Traducido al castellano debería sonar como "lenguaje intérprete cosido/entrelazado".

- Un compilador ANP (Anotación Polaca Inversa).

- Una librería C68 llamada OLIMPO, y escrita por Pedro Reina, que aporta un sub-lenguaje 'C' orientado al objeto.

Además de estas tres cosas he tenido que añadir una serie de estructuras y palabras extras para ayudar a la compilación.

Adjunto código objeto y fuente (incluido LIB OLIMPO v1.0), y un programa 'Demo' escrito en TIL OLIMPO para demostrar sus posibilidades ('load_file til_demo_olp'). El fichero TIL_HELP_OLP contiene la ayuda. Como es un fichero ASCII normal podéis imprimirlo o verlo en cualquier editor, y no olvidéis que podéis consultar cualquier comando tecleando 'help help'.

Si miráis con detenimiento los ficheros Demos terminados en _OLP, os daréis cuenta de que programar en TIL OLIMPO es tan sencillo como hacerlo directamente en 'C68', solamente hay que tener en cuenta que todo se escribe al revés (Anotación Polaca Inversa). Por ejemplo:

```
 : hola_mundo " Hola. Soy Olimpo" pan_texto ;  <ENTER>

 hola_mundo  <ENTER>
```

Esto último habría sido usando los comandos OLIMPO en modo compilación. En modo FORTH habría sido asi:

```
  : hola_mundo ." Hola. Soy Olimpo" ;  <ENTER>

  hola_mundo <ENTER>
```

Pero si lo hubiésemos hecho en modo intérprete habría sido así:

```
   40 $variable HOLA_MUNDO     <ENTER>
   HOLA_MUNDO read" Hola. Soy Olimpo"  <ENTER>
   HOLA_MUNDO pan_texto    <ENTER>
```

Actualmente estoy estudiando el libro 'Concepción y Diseño de Bases de Datos' (Ra-ma) de 1081 páginas. Mi próximo objetivo es convertir TIL OLIMPO en un poderoso lenguaje de base de datos.

A pesar de que TIL OLIMPO se suministra con ficheros fuente, he de advertir que se trata de un programa shareware, pues mis futuros programas estarán basados en él. Sin embargo, doy autorización a cualquier socio de QLíper para que haga uso de cualquier rutina en sus programas.

Espero que TIL OLIMPO, el cual es un lenguaje orientado al objeto y de cuarta generación (FORTH fue el primer lenguaje de cuarta generación, y de ahí proviene su nombre), tenga la aceptación que se merece.

Salvador Merino, 8/9/1993

WRITE YOUR OWN PROGRAMMING LANGUAGE USING C++

Este libro se puede comprar por mediación de FIG (Forth Interest Group, P.O. BOX 2154, OAKLAND, CALIFORNIA 94621, USA). Item# 270, Unit price $15 and postage $6. En total $21. Se puede pagar por mediación de VISA.

Lo que vais a recibir es un libro de 108 páginas en inglés y un disco 5.25" de 360K en formato MS-DOS conteniendo los ficheros fuente del programa CALC compilables bajo TURBO C++, y el ejecutable CALC.EXE

Se trata de un libro muy interesante que explica con muchos detalles e incluso suministra todas las rutinas necesarias para escribir nuestro propio lenguaje TIL (Threaded Interpretive Language) en 'C'.

El código fuente C++ no era ANSI C STANDARD. He tenido que hacer muchísimas modificaciones (¡unas 8 horas de trabajo!) al código para poder compilarlo bajo C68. Curiosamente, el nuevo código fuente es compilable bajo Turbo C++ alterando 4 líneas de 2.000 que tiene el programa.

Una curiosidad, el programa objeto o ejecutable CALC.EXE (MS-DOS) tiene un tamaño casi 3 veces superior al programa objeto o ejecutable QDOS. Esto me hace pensar que nuestro compilador C68 genera un código muchísimo más eficiente que el TURBO C++, pues si hay que ejecutar tres veces menos instrucciones para realizar el mismo trabajo, por fuerza debemos ser más rápidos en igualdad de condiciones (microprocesadores de similares características).

El programa CALC es muy sencillo de utilizar. Prácticamente os va a recordar a un lenguaje FORTH, pues CALC fue diseñado para escribir un lenguaje FORTH. Lo primero que hace el programa es compilar un fichero llamado calc_app que contiene 3 nuevos comandos (hello, fib y factorial). El comando macros nos muestra todos los comandos que hay en el diccionario. El comando help busca la ayuda en el fichero help_app. Este fichero ayuda podéis verlo con cualquier editor de texto o pasarlo directamente a impresora, pues se trata de un fichero ASCII normal que contiene toda la información sobre los comandos definidos en el sistema como primitivos.

Adjunto ficheros fuente CALC_c para C68 y ejecutable EXEC calc_exe.

Con estas rutinas fuente es facilísimo y muy rápido crear cualquier lenguaje tipo TIL. El compilador sigue siendo del tipo ANP (Anotación Polaca Inversa), pues facilita y hace muy sencilla la compilación. Aunque también se puede alterar a anotación normal, pero complica mucho el código y hace más lenta la compilación.

Mis futuros proyectos son realizar una versión TIL del sistema OLIMPO de Pedro Reina, y una base de datos programable TIL basada en mi Foto-DBase.

Sin la ayuda de Forth Interest Group, jamás habría podido escribir mis propios lenguajes (Z88-FORTH en 1990, eFORTH para QDOS en 1992, CALC para QDOS en 1993...). Gracias FIG por enseñarme todo lo que tenía que saber sobre TILs.

Otra cosilla, el Salvadormerinés, un hito en la historia de los lenguajes de programación. Un lenguaje usando un revuelto de raíces latinas y germánicas. ¿Será algún día una realidad? Yo diría que nada podría hoy impedirlo salvo invalidez mental o muerte prematura del autor.

Lo más importante de todo. Estas rutinas 'C' son multiplaforma. Esto significa que nuestros futuros lenguajes TIL estarían disponibles en diferentes máquinas y sistemas operativos que dispongan de un compilador 'C' que siga el estandár ANSI. Esto es la ley del mínimo esfuerzo y la máxima compatibilidad.

Salvador Merino, 10/8/93

Especualaciones sobre Inteligencia Artificial

1. Introducción.

La inteligencia artificial ha sido desde hace tiempo un tema que me ha apasionado y al que he dedicado algunos ratos en mi pensamiento.

Siempre he leído todo lo que sobre estos temas ha caído en mis manos, pero no soy un experto ni he leído los libros más avanzados sobre estos temas, cosa que sería de mi gusto. Por tanto lo que se expone a continuación no deja de ser más que lo que en el título se adelantaba.

No hace mucho, un gran científico visitó este país y a mi parecer, desde una postura bastante egocéntrica del hombre, refutó los argumentos que defienden los que apuestan por la inteligencia artificial fuerte, como se ha denominado a la inteligencia artificial que trata de que una máquina pueda llegar a ser consciente e incluso, llegado a este punto, superar al hombre en todas las facetas de su entendimiento. Quizá la verdadera postura egocéntrica sea la mía, pero la postura de este científico me pareció más la idea de una persona que al final de su vida piensa más en la espiritualidad que en la Ciencia. Se apoyó en un complejo razonamiento por el cual demostraba que no existe un algoritmo que sea capaz de discernir si cualquier algoritmo llega a un final o no. Da por sentado que una máquina inteligente debe funcionar empleando algoritmos, de esos a los que tan acostumbrados estamos, exactos, precisos y concretos. Yo me pregunto, ¿es probable que el hombre sea capaz de hacer eso que se le exige a la máquina?, pienso que no, y por tanto no veo en esto un argumento sólido para negar su potencial de inteligencia.

2. Definiendo la inteligencia.

A.A. Berk en su libro "Lisp. El lenguaje de la inteligencia artificial" dice: 'No hay ninguna definición aceptada de esta escurridiza cualidad, aunque todos podamos reconocerla. Esto tiene una consecuencia importante: significa que cualquiera puede "chapotear" en el estudio de la inteligencia. No hay ninguna razón por la que el aficionado no sea capaz de hacer interesantes incursiones en el tema por su cuenta. El campo está abierto a todos'.

Aunque es un concepto que todo el mundo utiliza y cree saber en que consiste, la verdad es que se trata de algo abstracto y su definición entra en el campo de la relatividad del lenguaje humano. Se habla de tipos de inteligencia, para pasar a definir la inteligencia cualitativa y cuantitativamente. Aunque en muchos aspectos las máquinas llegan a ser cuantítativamente más inteligentes que el hombre, cualitativamente la inteligencia artificial no alcanza aún, ni por los tobillos, a la inteligencia humana. Esto es un hecho cierto al día de hoy, pero lo que no creo que se pueda hacer es afirmar que siempre lo será, o que es imposible que sea de otra manera.

Un ordenador es más rápido que el hombre haciendo cálculos matemáticos, y es lógico si pensamos que su estructura está diseñada para ello, y no por ello deja de ser una máquina tonta cuando realiza estas actividades.

Se ha ligado muchas veces la inteligencia humana a su lenguaje, se dice que se piensa con él, pero esto es muy discutible. El lenguaje es una herramienta, es un conjunto de símbolos y no el medio del pensamiento humano, que a mi entender son los conceptos y sus relaciones. Existe una rama de la inteligencia artificial que va por ese camino, por el análisis del lenguaje humano, y a partir de él conseguir su comprensión. Se alcanzarán grandes logros, pero no lo conseguirá por sí solo; será necesario ligarlo a los conceptos que no son solamente lenguaje, sino sentidos. En este sentido no comparto la opinión de quienes ligan la inteligencia a la capacidad de comunicación, y utilizan esta capacidad como una medida de ella.

La mayoría de los autores coinciden en que no existe una definición completa de inteligencia, y yo no la voy a dar, pero analicemos lo que es capaz de hacer el hombre con la que posee.

¿Puede un ciego de nacimiento saber lo que es el color rojo, por ejemplo?. Yo pienso que no y por lo mismo un ordenador de los actuales tampoco. Pueden explicárselo, hacerle comparaciones con otras experiencias que sí pueda sentir, pero no sentir lo que siente el que lo ve. Esto es mi opinión, que quizá debería preguntárselo a alguno

para confirmarlo. ¿Pueden llegar a sentir la belleza del crepúsculo un tarde de primavera arrebolada?. No podemos tratar de que un ordenador comprenda el significado de una palabra por sí misma, aunque a esto me cabe la pregunta, ¿puede una máquina llegar a comprender y a tener entendimiento?. No lo sé con certeza, no lo afirmaré, pero pienso que sí es posible.

Cuando un niño nace y comienza a tener contacto con su entorno todo es nuevo y en su cerebro sólo está escrita una estructura que le permitirá construir un complejo edificio de conocimientos y sensaciones. El niño comienza estableciendo relaciones entre las diferentes percepciones que le llegan desde fuera y desde dentro, esto último lo es desde su programa interno o instintivo. Así que una alta capacidad de relacionar todo lo que percibe le va dando la capacidad de comprensión de él mismo y de su entorno. Relaciona lo que le produce satisfacción y lo desea, lo que le produce dolor y lo detesta, aprende a relacionar ciertos sonidos con otras sensaciones de placer o de dolor, aprende a relacionar ciertos sonidos con unos determinados movimientos de su boca, finalmente comienza a articular palabras con cierto sentido, en definitiva a hablar, lo que le lleva a un aumento importante en la capacidad de establecer relaciones, pero en ninguna manera su inteligencia comienza en este punto. Según este discurso lo que verdaderamente define la inteligencia es la capacidad de establecer relaciones entre las diferentes percepciones tanto externas como internas, llamando ahora internas a aquellas percepciones que son el resultado de una elaboración por parte del sistema relacional que constituye el cerebro y que más que percepciones, podríamos llamar conceptos o ideas.

El que sea otra definición más de lo que es inteligencia, más o menos completa, no importa mucho, lo importante es que nos aproximemos a la verdad sobre cómo funciona nuestro cerebro, y sobre esto quizá otros más que yo puedan opinar, me refiero a psicólogos y neurólogos. Pero yo desde mi experiencia personal puedo decir que mis actos y mis pensamientos siempre los encamino hacia mi satisfación directa o indirectamente. Cuando tengo que tomar una decisión evalúo consciente o incoscientemente y decido. El santo que se sacrifica y hace ayunos actúa así porque le produce satisfacción, aunque no física, sí espiritual. La persona que arriesga su vida para salvar a un amigo, en un acto de altruismo, en el momento de decidir hacerlo pesaron más los argumentos que le incitaban a que así actuara, que aquellos que indicaban lo contrario, aunque lo hiciera sin pensarlo de forma consciente.

¿Y qué puedo deciros que entiendo por pensamiento?. Yo en estado de vigilia, esto es, cuando estoy despierto, no puedo parar de pensar aunque quiera, y ¿no voy a saber lo que es en realidad?, os lo diré: una continua evaluación de lo que percibo tanto interior como exteriormente a través de los sentidos, estableciendo relaciones, y entre ellas, relaciones de respuesta a través de mis órganos motrices, incluida la voz y la escritura.

Y ¿si estáis de acuerdo, los que leéis estas líneas, con lo expuesto anteriormente?, quizá tengáis más claro ahora el concepto de inteligencia humana aunque sigamos sin saberlo definir completamente.

3. Aplicaciones de la inteligencia artificial.

En la actualidad, las aplicaciones de la inteligencia artificial se centran principalmente en lo que se suele llamar "sistemas expertos". Estos programas permiten la toma de decisiones a aquellos que los manejan, y básicamente consisten en un conjunto de información ordenada que mediante algoritmos relacionales permite llegar a soluciones correctas de problemas, ya resueltos anteriormente por un experto, dentro del área concreta que trate.

Existen programas que facilitan la creación de los sistemas expertos, y puedo referirme a un seminario al que asistí hará un año que versaba sobre uno de ellos. No dejé de verle muchas limitaciones, pues lo que

hacía este programa era facilitar la creación de la base de datos y de los algoritmos de decisión necesarios, pero todo ello era factible de hacer con cualquier lenguaje de programación elemental (BASIC, por ejemplo). Añadía la posibilidad de la toma de decisiones con márgenes de confianza expresados en forma de porcentajes de probabilidad para las diferentes soluciones. Esto es algo que es fácil de programar en los lenguajes básicos, si bien son necesarios ciertos conocimientos estadísticos.

Por otra parte contamos entre las aplicaciones de la inteligencia artificial con aquellos programas que emplan la información que van procesando para ir modificando sus criterios de análisis. De estos programas podemos decir verdaderamente que aprenden, aunque dentro de algoritmos programados muy rígidos. Entre ellos puedo citar los programas que permiten descifrar mensajes codificados, basándose en la frecuencia de las letras dentro de un determinado idioma. Estos programas consiguen más éxito conforme procesan más texto del idioma en cuestión.

Otra posible aplicación que se me ocurre puede ser un programa para hacer pronósticos quinielísticos, que utilice los resultados que se van obteniendo en las diferentes jornadas de liga para ir modificando la importancia concedida a los diferentes factores que considere en la formulación del pronóstico de la siguiente jornada. La ventaja de este programa es la existencia de un volumen de datos enorme con el que se pueden contrastar las diferentes predicciones. Un determinado modelo, o función evaluadora, puede ser optimizada de forma que proporcione un elevado porcentaje de aciertos, si bien hay que dejar claro que el hecho de que en el pasado se haya seguido una determinada pauta no quiere decir que se tenga que seguir, necesariamente, en el futuro. Lo que sí se ha de fijar previamente será el modelo, esto es la forma de la función evaluadora, sea cierta o no, y el programa ajustará los coeficientes de sus diferentes términos de forma automática. Este ejemplo lo he mencionado, pues hace tiempo realicé un pequeño intento de algo parecido. Eran mis tiempos de COMODORE 64, con todas las limitaciones en cuanto al manejo de los datos que suponía. Los datos se almacenaban en el propio listado de forma automática, pero la evaluación del modelo propuesto era demasiado lenta. A pesar de ello los resultados se mostraban prometedores. Este programa lo he pasado a SuperBasic mejorando considerablemente la organización de los datos y su almacenamiento, que ahora lo hace en disco, pero a pesar de ello aún me queda por definir un modelo adecuado en función de variables, como clasificación en la liga, goles encajados o realizados, resultados de los últimos partidos, y no olvidando una componente ciertamente aleatoria. ¿Será aditivo, multiplicativo, potencial, exponencial o combinación de todos o algunos de ellos?, no lo sé, no lo he decidido. Luego quedará programar la forma de automatizar la optimización de los coeficientes que intervengan. Ciertamente será un programa que "aprenda" en alguna medida. Espero en próximas colaboraciones proporcionar una copia del programa, aún rudimentario.

Pero la inteligencia artificial no debe estancarse en estas aplicaciones, deberá ir más allá. No será suficiente con que el programa evalúe uno o varios modelos propuestos, habrá que ir a que él mismo los elabore y proponga, los utilice resolviendo el problema exclusivamente a partir de la información que tenga disponible y unas normas elementales en forma de algoritmos evaluadores de los resultados propuestos. La máquina sea el auténtico experto que se programa a sí misma, desarrollándose a partir de unas normas que podíamos decir cuasi-instintivas, preprogramadas.

Podrá ser o no que se desarrollen máquinas de inteligencia similar a la humana, eso ya se verá más tarde o más temprano. La principal dificultad que quizá se encuentre, será el alcanzar densidades de memoria similaraes a las existentes en el cerebro humano, la máquina más prodigiosa.

Sin aspirar a esos niveles elevados de inteligencia, sí será posible conseguir imitar inteligencias menos desarrolladas, quizá empezando por aquellos animales que se rigen por un programa genético rígido

manifestado en sus fuertes instintos.

Espero que la lectura os haya resultado amena e interesante, y que os haya despertado opiniones contrarias o favorables que me gustaría conocer en cualquier caso.

En una próxima colaboración contaré algunas ideas que tengo de cómo se podría diseñar una máquina inteligente y cómo se puede desarrollar un programa que pueda aprender por sí solo. Sin nada más que añadir y esperando que esta que es mi primera colaboración no sea la última, me despido, si bien antes os indico mi nuevo domicilio.

Rafael Illanes Muñoz
c/Salvador Dalí, 4 -Esc. Dcha.-1B
28933 MOSTOLES
Madrid

En este disco hay una serie de programas SuperBasic que había desarrollado hace algún tiempo. He incluido en los propios listados de los programas los comentarios explicativos. El programa "transferir" deberá usarse con precaución, pues puede producir pérdidas de datos en el caso de que el disco en la unidad flp2_ no sea el apropiado, al realizar la escritura de datos sin tener en cuenta el mapa de direcciones de ficheros.

Ultimamente estoy muy interesado en el lenguaje C, especialmente en el uso del C68, con el que he tenido algunos problemas al no disponer aún de la última versión. También me ha parecido muy interesante el C++ de Borlan, en el que me ha sorprendido la posibilidad de incluir funciones dentro de las diferentes estructuras de datos como parte integrante de las mismas. La potencia que puede dar a la programación el hecho de que los objetos queden completamente definidos por sus datos descriptivos y sus relaciones funcionales, dentro de la propias estructuras, pienso que supone un gran avance para el futuro de la programación, y que deberá ser objeto de un estudio más detenido por mi parte.

Móstoles, 4 de Octubre de 1993.

Rafael Illanes Muñoz.